日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　３月２８日

出願番号
Application Number: 特願２００３－０９２３７１

[ST. 10/C]: ［ＪＰ２００３－０９２３７１］

出願人
Applicant(s): セイコーエプソン株式会社

２００４年　１月２６日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００４－３００２７００

【書類名】　特許願
【整理番号】　J0097495
【提出日】　平成15年 3月28日
【あて先】　特許庁長官殿
【国際特許分類】　H04L 12/00
【発明者】
　【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内
　【氏名】　礒村　政一
【特許出願人】
　【識別番号】　000002369
　【氏名又は名称】　セイコーエプソン株式会社
【代理人】
　【識別番号】　100066980
　【弁理士】
　【氏名又は名称】　森　哲也
【選任した代理人】
　【識別番号】　100075579
　【弁理士】
　【氏名又は名称】　内藤　嘉昭
【選任した代理人】
　【識別番号】　100103850
　【弁理士】
　【氏名又は名称】　崔　秀▲てつ▼
【手数料の表示】
　【予納台帳番号】　001638
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0014966

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ベクトルプロセッサおよびレジスタのアドレス指定方法

【特許請求の範囲】

【請求項１】　複数の要素データからなるベクトルデータを、レジスタを使用して演算処理するベクトルプロセッサであって、

複数の要素レジスタからなるベクトルレジスタとして使用可能なレジスタと、

該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として、前記ベクトルレジスタのアドレスを、循環的に指定するアドレス指定回路と、

を備えることを特徴とするベクトルプロセッサ。

【請求項２】　前記レジスタは、複数のスカラーレジスタが１組となり、該１組のスカラーレジスタのいずれかのアドレスが先頭として指定されることにより、該複数のスカラーレジスタそれぞれのアドレスが循環的に指定されることを特徴とする請求項１記載のベクトルプロセッサ。

【請求項３】　前記レジスタは、任意の要素レジスタを先頭として指定可能なベクトルレジスタによって構成されていることを特徴とする請求項１記載のベクトルプロセッサ。

【請求項４】　前記レジスタに記憶されたデータを対象としてベクトル演算を行う際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次読み出し、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの読み出しを継続可能であることを特徴とする請求項１～３のいずれかに記載のベクトルプロセッサ。

【請求項５】　ベクトル演算の演算結果を前記レジスタに書き込む際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次書き込み、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの書き込みを継続可能であることを特徴とする請求項１～４のいずれかに記載のベクトルプロセッサ。

【請求項６】　複数の要素データからなるベクトルデータの演算処理に用いるレジスタのアドレス指定方法であって、

所定の要素レジスタを複数の要素レジスタからなるベクトルレジスタとして取り扱い、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として指定すると、前記ベクトルレジスタにおける各要素レジスタのアドレスが循環的に指定されることを特徴とするレジスタのアドレス指定方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ベクトルレジスタを使用して演算を行うベクトルプロセッサおよびレジスタのアドレス指定方法に関する。

【０００２】

【従来の技術】

従来、科学技術計算における繰り返し演算や、画像処理における画素データの演算あるいは配列データの演算等において、ベクトル演算が行われている。

ベクトル演算においては、メモリからベクトルデータを読み出し、ベクトルレジスタに記憶した上で、ベクトルデータ同士の加算あるいは乗算等、所定のベクトル演算が行われる。

【０００３】

例えば、配列において隣接する２つのデータを対象として、加算等の演算を行う場合、１つのベクトルレジスタに配列データを先頭からｎ個（ｎは自然数）分記憶し、他のベクトルレジスタには、配列データを２番目のものからｎ個分記憶する。そして、それら２つのベクトルレジスタを対象として加算等の演算命令を実行することにより、ベクトルレジスタ内における同一アドレス同士の各要素データが演算され、配列データの演算が一括して行われる。

【０００４】

ここで、ベクトル演算に関連する技術として、特開昭６０－２４６７２号公報に記載された技術が知られている。

本号公報においては、上述のように、配列において隣接する２つのデータを対象として演算を行う際、演算処理における効率を向上させる技術が開示されている。

【０００５】

即ち、上述の演算を行う場合、配列の２番目からｎ－１個（演算される配列データの末尾の１つ前）のデータは、２つのレジスタに記憶するために、それぞれ２度ずつ読み出されることとなる。

そこで、本号公報に記載された技術では、演算対象となる配列を先頭からｎ＋１個（演算される配列データの末尾）まで一度だけ読み出し、所定のレジスタに記憶する。そして、そのレジスタ内のデータを複数のベクトルレジスタに分配して記憶させる分配回路によって、先頭から所定数分の配列データを格納するベクトルレジスタと、２番目のものから所定数分格納するベクトルレジスタとに振り分けて記憶する。

【０００６】

このような手順とすることにより、メモリから配列データを読み出す際に、同一のデータが重複して読み出されることを回避し、効率的にベクトル演算を行おうとするものである。

【０００７】

【特許文献１】

特開昭６０－２４６７２号公報

【０００８】

【発明が解決しようとする課題】

しかしながら、特開昭６０－２４６７２号公報に記載された技術においては、メモリから読み出したベクトルデータを一旦、所定のレジスタに記憶した後に、２つのベクトルレジスタに分配するため、多くのレジスタリソースが必要となる。

【０００９】

したがって、回路規模が大きくなるという問題や、レジスタリソースが圧迫され、処理効率が低下するという問題が生じていた。

さらに、近年、機器の低消費電力化が重要視されており、書き込みを行う場合にのみ、レジスタを構成するフリップフロップに電力を供給することが一般的である。一方、特開昭６０－２４６７３号公報に記載された技術においては、同一

のデータを同時に複数のベクトルレジスタに記憶することとなるため、同時にクロックを供給する対象が増え、消費電力が増大するという問題があった。

【００１０】

本発明の課題は、ベクトルレジスタを用いたベクトル演算を効率的に行うことである。

【００１１】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

複数の要素データからなるベクトルデータを、レジスタを使用して演算処理するベクトルプロセッサであって、複数の要素レジスタからなるベクトルレジスタとして使用可能なレジスタ（例えば、図６のレジスタファイル４０）と、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として、前記ベクトルレジスタのアドレスを、循環的に指定するアドレス指定回路（例えば、図１０に示す第１ソースレジスタ決定回路７２等）とを備えることを特徴としている。

【００１２】

また、前記レジスタは、複数のスカラーレジスタが１組となり、該１組のスカラーレジスタのいずれかのアドレスが先頭として指定されることにより、該複数のスカラーレジスタそれぞれのアドレスが循環的に指定されることを特徴としている。

また、前記レジスタは、任意の要素レジスタを先頭として指定可能なベクトルレジスタによって構成されていることを特徴としている。

【００１３】

また、前記レジスタに記憶されたデータを対象としてベクトル演算を行う際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次読み出し、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの読み出しを継続可能であることを特徴としている。

【００１４】

また、ベクトル演算の演算結果を前記レジスタに書き込む際、前記ベクトルレ

ジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次書き込み、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの書き込みを継続可能であることを特徴としている。

【００１５】

また、本発明は、

複数の要素データからなるベクトルデータの演算処理に用いるレジスタのアドレス指定方法であって、所定の要素レジスタを複数の要素レジスタからなるベクトルレジスタとして取り扱い、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として指定すると、前記ベクトルレジスタにおける各要素レジスタのアドレスが循環的に指定されることを特徴としている。

【００１６】

本発明によれば、リングバッファをなすベクトルレジスタとして使用可能なレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。

そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、１つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むことができる。

【００１７】

したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【００１８】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

本発明に係るベクトルプロセッサは、リングバッファをなすベクトルレジスタを備え、ベクトルレジスタの任意のアドレスからデータにアクセスすることを可

能としている。

【００１９】

したがって、初めに、本発明の基本となる考え方について説明する。なお、ここでは、配列において隣接する２つのデータを対象として加算を行う場合（例えば、画像処理において、隣接画素同士の平均値を求める処理）を例に挙げて説明する。

図１は、８つの要素レジスタＲ０～Ｒ７を有するベクトルレジスタＶＲを示す図である。

【００２０】

本発明においては、加算対象となるベクトルデータを記憶するベクトルレジスタを２つ用いることなく、１つのベクトルレジスタＶＲによって、２つのベクトルレジスタを用いる場合と同様の処理を行う。なお、ここでいうベクトルレジスタには、スカラーレジスタをベクトルレジスタとして使用する場合を含むものである。

【００２１】

まず、ベクトルレジスタＶＲを備えるベクトルプロセッサに、８つの要素データｘ０～ｘ７からなるベクトルデータのロード命令と、そのロード命令から２サイクル遅れた加算命令とを与える。

すると、サイクル“０”において、ベクトルレジスタＶＲの要素レジスタＲ０に要素データｘ０が書き込まれ、１サイクル毎に、引き続く要素レジスタに後続の要素データが順次書き込まれていく。

【００２２】

ここで、サイクル“２”において、サイクル“０”に開始されたロード命令から２サイクル遅れて、加算命令が実行される。

図２は、サイクル“２”におけるベクトルレジスタＶＲの状態を示す図である。

図２において、要素レジスタＲ０，Ｒ１には、それぞれ要素データｘ０，ｘ１が既に記憶されており、さらに要素レジスタＲ２には、要素データｘ２が書き込まれている。また、図２においては、加算命令が開始されており、要素レジスタ

Ｒ０，Ｒ１に記憶された要素データが加算されている。

【００２３】

このように、ロード命令から２サイクル遅れて加算命令を実行していくと、サイクル“７”においては、図３に示す状態となる。

図３において、ロード命令については、要素レジスタＲ７に要素データｘ７を書き込む状態であり、サイクル“０”で開始されたロード命令はサイクル“７”で終了する。なお、加算命令については、要素レジスタＲ５，Ｒ６に記憶された要素データが加算されている。

【００２４】

次に、ベクトルプロセッサには、後続のデータを処理するため、サイクル“８”において、２つ目のロード命令が与えられる。

すると、サイクル“８”においては、ベクトルレジスタＶＲは、図４に示す状態となる。

図４において、ロード命令については、先頭に戻り、要素レジスタＲ０に要素データｘ８を書き込む状態であり、加算命令については、要素レジスタＲ６，Ｒ７に記憶された要素データが加算されている。なお、サイクル“２”で開始された加算命令は、ロード命令から２サイクル送れているため、依然として加算処理を実行している。

【００２５】

続いて、サイクル“２”で開始された加算命令の８サイクル目（最終サイクル）であるサイクル“９”に移行する。

図５は、サイクル“９”におけるベクトルレジスタＶＲの状態を示す図である。

図５において、加算命令については、要素レジスタＲ７から、加算対象の１つとなる要素データｘ７が読み出されている。

【００２６】

ここで、加算対象となる他の要素データについては、ベクトルレジスタＶＲの先頭に戻り、要素レジスタＲ０に記憶されている要素データｘ８が読み出されている。即ち、要素レジスタＲ７，Ｒ０に記憶された要素データｘ７，ｘ８が加算

される。

なお、この後、サイクル“０”～“９”が適宜繰り返される。

【００２７】

このように、加算命令によって、ベクトルレジスタの最終アドレスを超えた要素データが参照される場合であっても、ベクトルレジスタがリングバッファをなすことによって、参照される要素データを容易に読み出すことができ、引き続く処理を円滑に行うことができる。

また、同一の要素データをメモリ等から重複して読み出す必要がなく、また、８要素を記憶可能なベクトルレジスタを１つ用いることで、８要素を超えるベクトルデータの要素データ同士の演算を行うことができるため、ベクトル演算を効率的に行うことが可能となる。

【００２８】

次に、本発明に係るベクトルプロセッサの構成を説明する。

図６は、本発明を適用したベクトルプロセッサ１の構成を示す図である。

図６において、ベクトルプロセッサ１は、メモリ１０と、メモリ制御部２０と、命令フェッチ部３０と、レジスタファイル４０と、ロードユニット５０と、ストアユニット６０と、演算ユニット７０とを含んで構成される。

【００２９】

メモリ１０は、ベクトルプロセッサ１に与えられる命令コードおよび演算対象となるデータを記憶している。

図７は、命令コードのデータ形式の一例を示す図であり、（ａ）は、ロード命令の形式、（ｂ）は、ストア命令の形式、（ｃ）は、演算命令の形式を示している。図７において、各命令コードには、命令コードの命令の種類を示すオペレーションコード、命令の処理対象となるベクトルデータの要素数あるいはレジスタの指定コード等、命令を実行するために必要となる情報が含まれている。

【００３０】

メモリ制御部２０は、メモリ１０に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部２０は、ロードユニット５０あるいはストアユニット６０によって指定されたメモリ１０のアドレスからデータを

読み出したり、メモリ１０から読み出されたデータをレジスタファイル４０に出力したりする。

【００３１】

命令フェッチ部３０は、メモリ制御部２０を介して、メモリ１０から命令コードをフェッチし、一時的に記憶する。

レジスタファイル４０は、メモリ１０から読み出されたデータおよび演算結果を一時的に記憶する。

ロードユニット５０は、命令フェッチ部３０に記憶された命令コードがロード命令である場合に、メモリ１０から命令コードあるいはデータを読み出す処理を行う。

【００３２】

ストアユニット６０は、命令フェッチ部３０に記憶された命令コードがストア命令である場合に、メモリ１０にデータを書き込む処理を行う。

演算ユニット７０は、命令フェッチ部３０に記憶された命令コードが所定の演算命令である場合に、レジスタファイル４０に記憶された所定データを対象として演算処理を行う。

【００３３】

ここで、レジスタファイル４０および演算ユニット７０について、詳細に説明する。

まず、レジスタファイル４０について説明する。

レジスタファイル４０は、図６に示すように、読み書き可能な３２個のレジスタＲ０～Ｒ３１を含んで構成される。

【００３４】

また、レジスタファイル４０において、レジスタＲ０～Ｒ７、レジスタＲ８～Ｒ１５、レジスタＲ１６～Ｒ２３およびレジスタＲ２４～Ｒ３１は、それぞれを１組として、リングバッファの機能を有するベクトルレジスタとして使用することが可能である。

ここで、リングバッファの機能を有するベクトルレジスタとして、レジスタＲ０～Ｒ３１を使用可能とするためには、ベクトルレジスタにリングバッファの機

能を備え、任意のアドレスを先頭アドレスとして指定可能とすることの他、スカラーレジスタをベクトルレジスタとして用いることが可能である。

【００３５】

即ち、所定のスカラーレジスタを１組とし、組単位でレジスタを指定可能とすることにより、スカラーレジスタをベクトルレジスタとして使用することが可能である。なお、この場合、１組のスカラーレジスタにおける任意のアドレスを先頭アドレスとして指定することができると共に、スカラーレジスタは本来、レジスタ毎の指定が可能であることから、リングバッファとして、循環的にアドレスを指定することができる。

【００３６】

また、図６に示すレジスタファイル４０において、レジスタＲ０～Ｒ３１には、それぞれに５ビットのコードが割り当てられている。

図８は、レジスタＲ０～Ｒ３１それぞれに割り当てられたコードを示す図である。

レジスタファイル４０に対し、図８に示す所定のコードを選択信号として入力することにより、対応するレジスタの読み出しおよび書き込みを行うことができる。

【００３７】

なお、図８において、５ビットのコードのうち、上位２ビットはベクトルレジスタを指定するコードであり、下位３ビットは、ベクトルレジスタ内のアドレスを指定するコードである。

続いて、演算ユニット７０について説明する。

図９は、演算ユニット７０の内部構成を示すブロック図である。

【００３８】

図９において、演算ユニット７０は、命令パイプライン制御部７１と、第１ソースレジスタ決定回路７２と、第２ソースレジスタ決定回路７３と、デスティネーションレジスタ決定回路７４と、演算器７５と、パイプラインレジスタ（ＰＲ）７６～７９とを含んで構成される。

命令パイプライン制御部７１は、演算ユニット７０全体を制御するものである

。

【００３９】

第１ソースレジスタ決定回路７２は、命令コードに含まれる第１ソースレジスタ指定コードに基づいて、第１ソースレジスタを選択する信号（第１ソースレジスタ選択信号）を生成する。

第２ソースレジスタ決定回路７３は、命令コードに含まれる第２ソースレジスタ指定コードに基づいて、第２ソースレジスタを選択する信号（第２ソースレジスタ選択信号）を生成する。

【００４０】

デスティネーションレジスタ決定回路７４は、命令コードに含まれるデスティネーションレジスタ指定コードに基づいて、デスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

ここで、第１ソースレジスタ決定回路７２、第２ソースレジスタ決定回路７３およびデスティネーションレジスタ決定回路７４の構成について説明する。

【００４１】

なお、第１ソースレジスタ決定回路７２、第２ソースレジスタ決定回路７３およびデスティネーションレジスタ決定回路７４の構成は同様であるため、第１ソースレジスタを例に挙げて説明する。

図１０は、第１ソースレジスタ決定回路７２の構成例を示す図である。

図１０において、第１ソースレジスタ決定回路７２は、制御部７２ａと、セレクタ７２ｂと、インクリメンタ７２ｃと、カウンタ７２ｄと、レジスタ７２ｅとを含んで構成される。

【００４２】

制御部７２ａは、命令パイプライン制御部７１によって入力される動作開始信号と、命令フェッチ部３０によって入力されるベクトル要素数に基づいて、第１ソースレジスタ決定回路７２全体を制御する。

セレクタ７２ｂは、サイクル“０”においては、命令フェッチ部３０から入力された第１ソースレジスタ指定コードを選択して出力し、サイクル“０”以外においては、カウンタ７２ｄおよびレジスタ７２ｅから入力された第１ソースレジ

スタ選択信号を選択して出力する。

【００４３】

インクリメンタ７２ｃは、５ビットの第１ソースレジスタ指定コードのうち、下位３ビットを受け取り、“１”加算してカウンタ７２ｄに出力する。

カウンタ７２ｄは、サイクル“０”において、インクリメンタ７２ｃから入力された３ビットのコードを記憶する。

また、カウンタ７２ｄは、制御部７２ａの指示によってカウントイネーブル状態が切り替えられ、カウントイネーブル状態である場合に、記憶しているコードを＋１して更新するカウントアップ動作を行う。

【００４４】

レジスタ７２ｅは、第１ソースレジスタ指定コードのうち、上位２ビットを受け取り、１つのベクトル演算が行われている間、そのコードを保持する。

このような構成により、第１ソースレジスタ決定回路７２においては、例えば、第１ソースレジスタ指定コードがレジスタＲ１８を示すコード“１００１０”であり、ベクトル要素数が“８”である場合、第１ソースレジスタ選択信号として、順次、“１００１０”、“１００１１”、“１０１００”、“１０１０１”、“１０１１０”、“１０１１１”が出力され、引き続き、“１００００”、“１００００１”が出力される。即ち、第１ソースレジスタ選択信号によって、レジスタＲ１８→レジスタＲ１９→レジスタＲ２０→レジスタＲ２１→レジスタＲ２２→レジスタＲ２３→レジスタＲ１６→レジスタＲ１７の順に選択される。

【００４５】

つまり、レジスタＲ１６～Ｒ２３を、リングバッファをなすベクトルレジスタとして使用できると共に、これらのレジスタにおける任意のアドレスを先頭アドレスとして指定することが可能となる。

図９に戻り、演算器７５は、命令パイプライン制御部７１の指示に基づいて、実際に加算等の演算を行う。

【００４６】

ＰＲ７６～７９は、パイプライン処理の各ステージにおいて処理されるデータを記憶する。

次に、動作を説明する。

初めに、図６を参照して、ベクトルプロセッサ１全体の動作について説明する。

【００４７】

ベクトルプロセッサ１において処理が行われる場合、メモリ制御部２０を介してメモリ１０から命令フェッチ部３０に命令コードが読み出される。

そして、ロードユニット５０、ストアユニット６０および演算ユニット７０のそれぞれに、命令フェッチ部３０から命令コードが出力される。

命令コードが入力されたロードユニット５０、ストアユニット６０および演算ユニット７０は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【００４８】

以下、命令コードの内容に分けて、動作を説明する。

（命令コードがロード命令である場合）

命令フェッチ部３０から入力された命令コードがロード命令である場合、ロードユニット５０は、命令コード（図７（ａ）参照）において指定された基底アドレスレジスタおよびアドレス修飾レジスタのそれぞれを選択する信号をレジスタファイル４０に出力する。

【００４９】

すると、それぞれのアドレスに記憶されている値（基底アドレス値およびアドレス修飾値）が、ロードユニット５０に読み込まれる。

そして、ロードユニット５０は、基底アドレス値およびアドレス修飾値に基づいて、メモリ１０のロードアドレス（読み込み対象アドレス）を生成し、メモリ制御部２０に出力する。

【００５０】

ロードアドレスが入力されると、メモリ制御部２０は、メモリ１０における対応するアドレスのデータ（ロードデータ）を読み出し、そのロードデータをレジスタファイル４０に出力する。このとき、ロードユニット５０は、メモリ制御部２０からレジスタファイル４０にロードデータが出力されるタイミングに合わせ

て、命令コードにおいて指定されているデスティネーションレジスタを選択するための信号を、レジスタファイル４０に出力する。

【００５１】

すると、レジスタファイル４０において、デスティネーションレジスタに、ロードデータが書き込まれる。

（命令コードがストア命令である場合）

命令フェッチ部３０から入力された命令コードがストア命令である場合、ストアユニット６０は、命令コード（図７（ｂ）参照）において指定されたデスティネーションレジスタ選択信号をレジスタファイル４０に出力する。

【００５２】

すると、デスティネーションアドレスに記憶されている値（ストアデータ）がストアユニット６０に読み込まれる。

そして、ストアユニット６０は、読み込んだストアデータをメモリ制御部２０に出力する。

また、ストアユニット６０は、命令コードにおいて指定されている基底アドレスレジスタおよびアドレス修飾レジスタのそれぞれを選択する信号をレジスタファイル４０に出力する。

【００５３】

すると、それぞれのアドレスに記憶されている基底アドレス値およびアドレス修飾値が、ストアユニット６０に読み込まれる。

そして、ストアユニット６０は、基底アドレス値およびアドレス修飾値に基づいて、メモリ１０のストアアドレス（書き込み対象アドレス）を生成し、ストアデータをメモリ制御部２０に出力するタイミングと合わせて、ストアアドレスをメモリ制御部２０に出力する。

【００５４】

ストアデータおよびストアアドレスが入力されると、メモリ制御部２０は、メモリ１０における対応するアドレスに、ストアデータを書き込む。

（命令コードが演算命令である場合）

命令フェッチ部３０から入力された命令コードが演算命令である場合、演算ユ

ニット７０は、命令コード（図７（ｃ）参照）において指定された第１ソースレジスタ選択信号および第２ソースレジスタ選択信号をレジスタファイル４０に出力する。

【００５５】

すると、それぞれのアドレスに記憶されている値（第１ソースデータおよび第２ソースデータ）が、演算ユニット７０に読み込まれる。

そして、演算ユニット７０は、第１ソースデータと第２ソースデータとの演算を行い、演算結果をレジスタファイル４０に出力する。このとき、演算ユニット７０は、演算結果がレジスタファイル４０に出力されるタイミングに合わせて、命令コードにおいて指定されているデスティネーションレジスタ選択信号を、レジスタファイル４０に出力する。

【００５６】

すると、レジスタファイル４０において、デスティネーションレジスタに、演算結果が書き込まれる。

次に、演算ユニット７０の動作について、図９を参照して詳細に説明する。

演算ユニット７０には、まず、命令フェッチ部３０からオペレーションコードおよびベクトル要素数が命令パイプライン制御部７１に入力される。

【００５７】

このとき入力されるベクトル要素数は、ベクトル演算を行う要素データ数を指定するコードであり、ここでは、図１１に示すように、３ビットのコードである。

命令フェッチ部３０から入力されたオペレーションコードが演算命令である場合、命令パイプライン制御部７１が、第１ソースレジスタ決定回路７２および第２ソースレジスタ決定回路７３に動作開始信号を出力する。

【００５８】

第１ソースレジスタ決定回路７２は、命令パイプライン制御部７１から動作開始信号が入力されると、命令フェッチ部３０から第１ソースレジスタ指定コードおよびベクトル要素数を受け取る。

そして、第１ソースレジスタ決定回路７２は、命令フェッチ部３０から受け取

ったベクトル要素数に応じて、所定のレジスタを選択するための第１ソースレジスタ選択信号をレジスタファイル４０に順次出力する。

【００５９】

すると、レジスタファイル４０から、所定の第１ソースデータが、ＰＲ７６に順次入力される。

また、第２ソースデータについても、第１ソースデータと同様の手順によって、ＰＲ７７に順次入力される。

そして、演算器７５が、ＰＲ７６，７７に記憶されている第１ソースデータおよび第２ソースデータの演算を行い、演算結果をレジスタファイル４０に出力する。

【００６０】

一方、命令フェッチ部３０から入力されたデスティネーションレジスタ指定コードは、ＰＲ７８に記憶され、同様にＰＲ７９に記憶されたベクトル要素数とタイミングを合わせられた上で、デスティネーションレジスタ決定回路７４に入力される。

すると、デスティネーションレジスタ決定回路７４は、演算器７５が演算結果をレジスタファイル４０に出力するタイミングに合わせて、所定のレジスタを選択するためのデスティネーションレジスタ選択信号をレジスタファイル４０に出力する。

【００６１】

このような手順を繰り返すことにより、レジスタファイル４０において、所定のデスティネーションレジスタに演算結果が順次書き込まれる。

以上のように、本実施の形態に係るベクトルプロセッサ１は、リングバッファをなすベクトルレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。

【００６２】

そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、１つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むこと

ができる。

したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【００６３】

また、同一のデータが重複して読み出されることを回避できるため、消費電力が軽減される。さらに、必要となるレジスタリソースが減少することから、回路規模を縮小できると共に、処理効率を向上させることができる。

なお、本実施の形態においては、第１ソースデータおよび第２ソースデータの両方について、リングバッファをなすベクトルレジスタに記憶して演算を行うこととして説明したが、いずれか一方のみをリングバッファをなすベクトルレジスタに記憶し、他方は一般的なベクトルレジスタあるいはスカラーレジスタに記憶して演算することとしてもよい。

【００６４】

また、本実施の形態においては、ベクトル要素数を命令コードによって指定することとして説明したが、レジスタファイル４０あるいは他のレジスタにベクトル要素数を格納し、そのレジスタを指定することとしてもよい。

【図面の簡単な説明】

【図１】　８つの要素レジスタＲ０～Ｒ７を有するベクトルレジスタＶＲを示す図である。

【図２】　サイクル“２”におけるベクトルレジスタＶＲの状態を示す図である。

【図３】　サイクル“７”におけるベクトルレジスタＶＲの状態を示す図である。

【図４】　サイクル“８”におけるベクトルレジスタＶＲの状態を示す図である。

【図５】　サイクル“９”におけるベクトルレジスタＶＲの状態を示す図である。

【図６】　本発明を適用したベクトルプロセッサ１の構成を示す図である。

【図７】　命令コードのデータ形式の一例を示す図である。

【図８】　レジスタＲ０～Ｒ３１それぞれに割り当てられたコードを示す図である。

【図９】　演算ユニット７０の内部構成を示すブロック図である。

【図１０】　第１ソースレジスタ決定回路７２の構成例を示す図である。

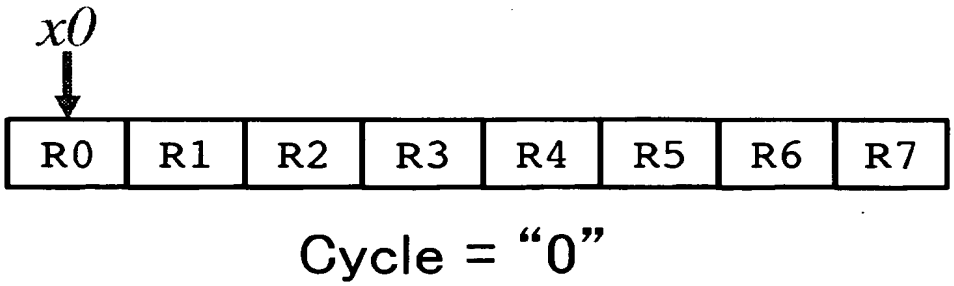
【図１１】　ベクトル要素数のコードを示す図である。

【符号の説明】

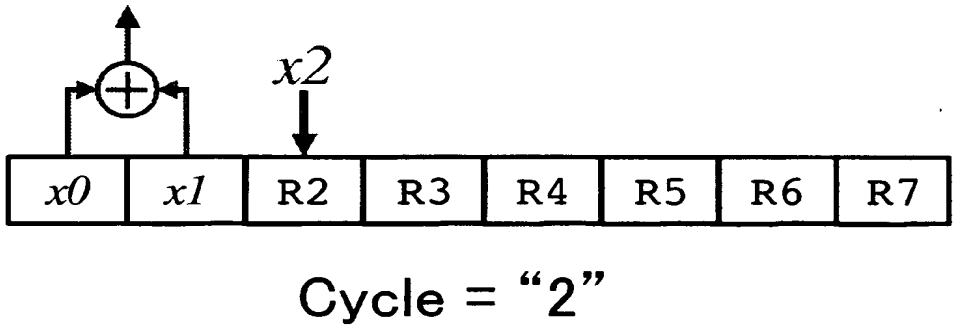
１　ベクトルプロセッサ，１０　メモリ，２０　メモリ制御部，３０　命令フェッチ部，４０　レジスタファイル，５０　ロードユニット，６０　ストアユニット，７０　演算ユニット，７１　命令パイプライン制御部，７２　ソースレジスタ決定回路，７２ａ　制御部，７２ｂ　セレクタ，７２ｃ　インクリメンタ　７２ｄ　カウンタ，７２ｅ　レジスタ，７３　ソースレジスタ決定回路，７４　デスティネーションレジスタ決定回路，７５　演算器，７６～７９　ＰＲ（パイプラインレジスタ）

【書類名】　図面

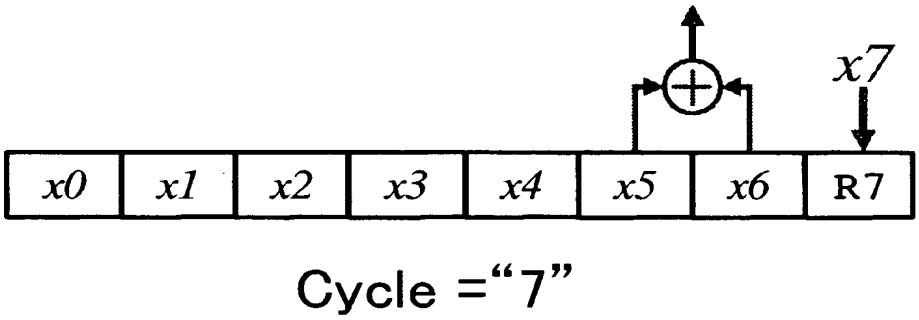
【図１】



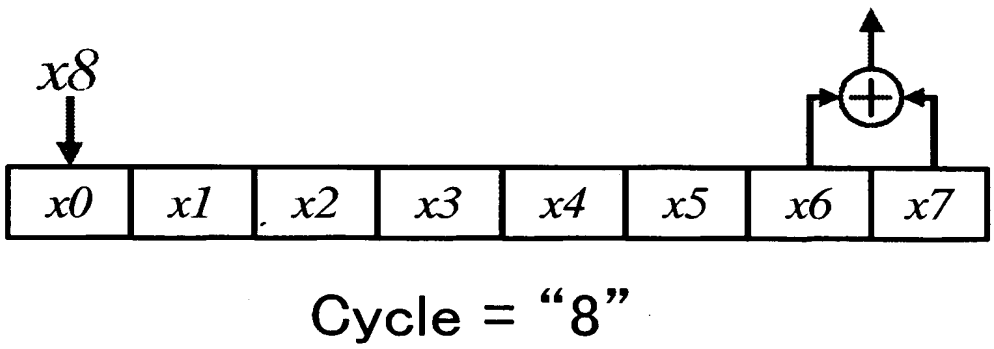
【図２】



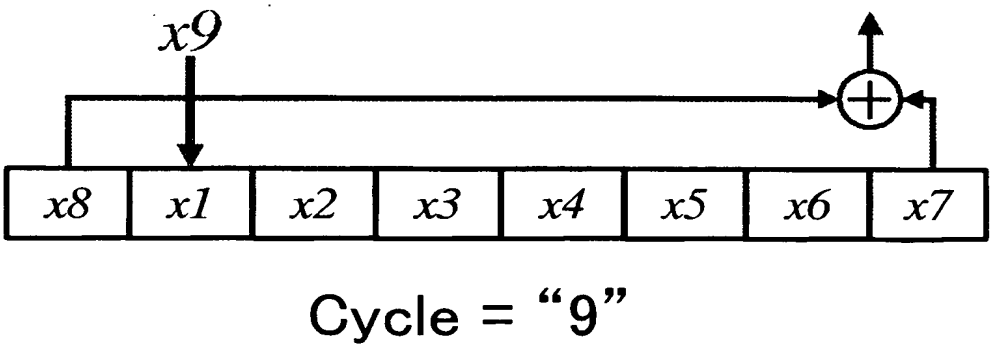
【図３】



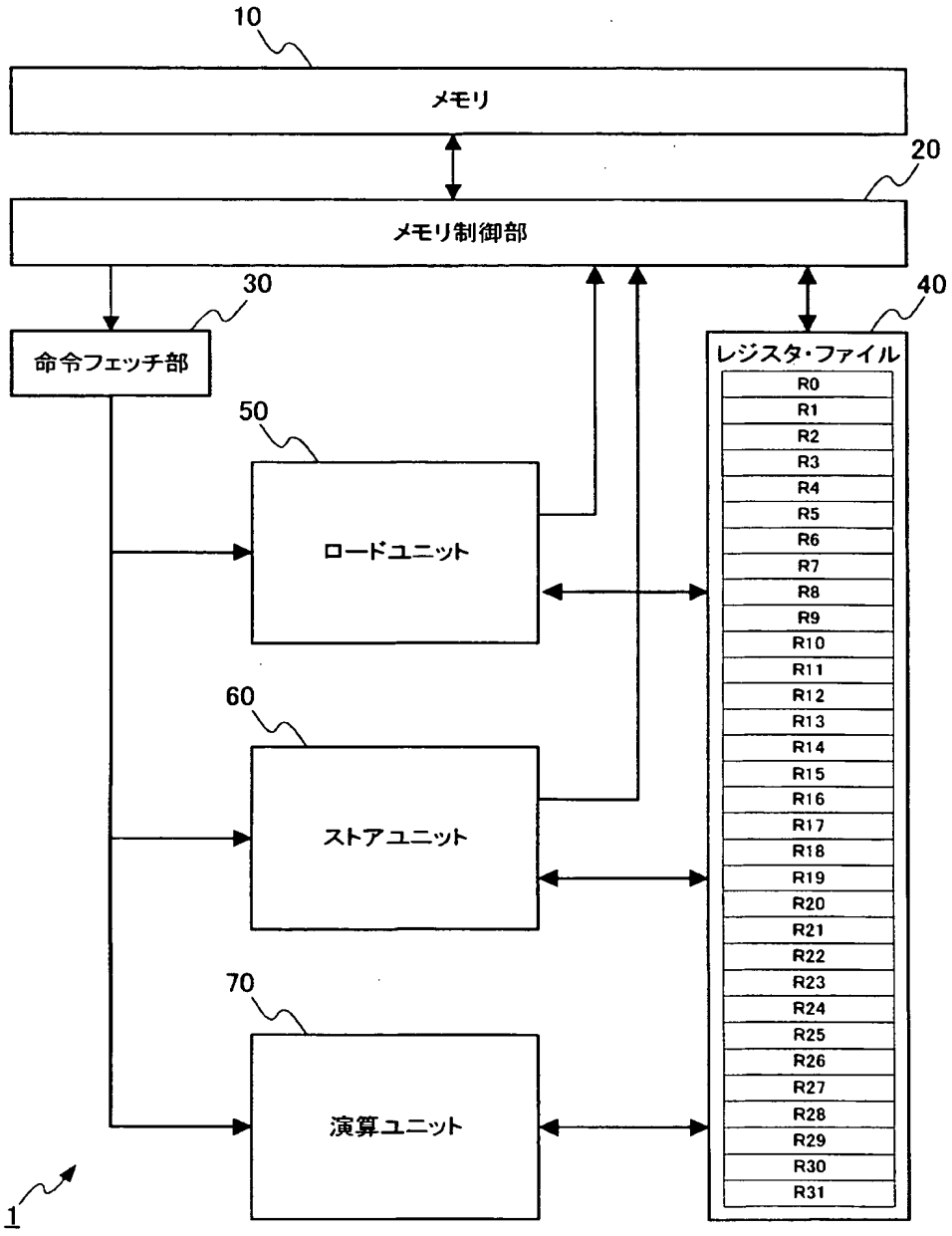
【図４】



【図５】



【図６】


10
メモリ
20
メモリ制御部
30
命令フェッチ部
40
レジスタ・ファイル
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
50
ロードユニット
60
ストアユニット
70
演算ユニット
1

【図７】

(a) ロード命令の形式

オペレーションコード	ベクトル要素数	ロードレジスタ指定コード	基底アドレスレジスタ指定コード	アドレス修飾レジスタ指定コード

(b) ストア命令の形式

オペレーションコード	ベクトル要素数	ストアレジスタ指定コード	基底アドレスレジスタ指定コード	アドレス修飾レジスタ指定コード

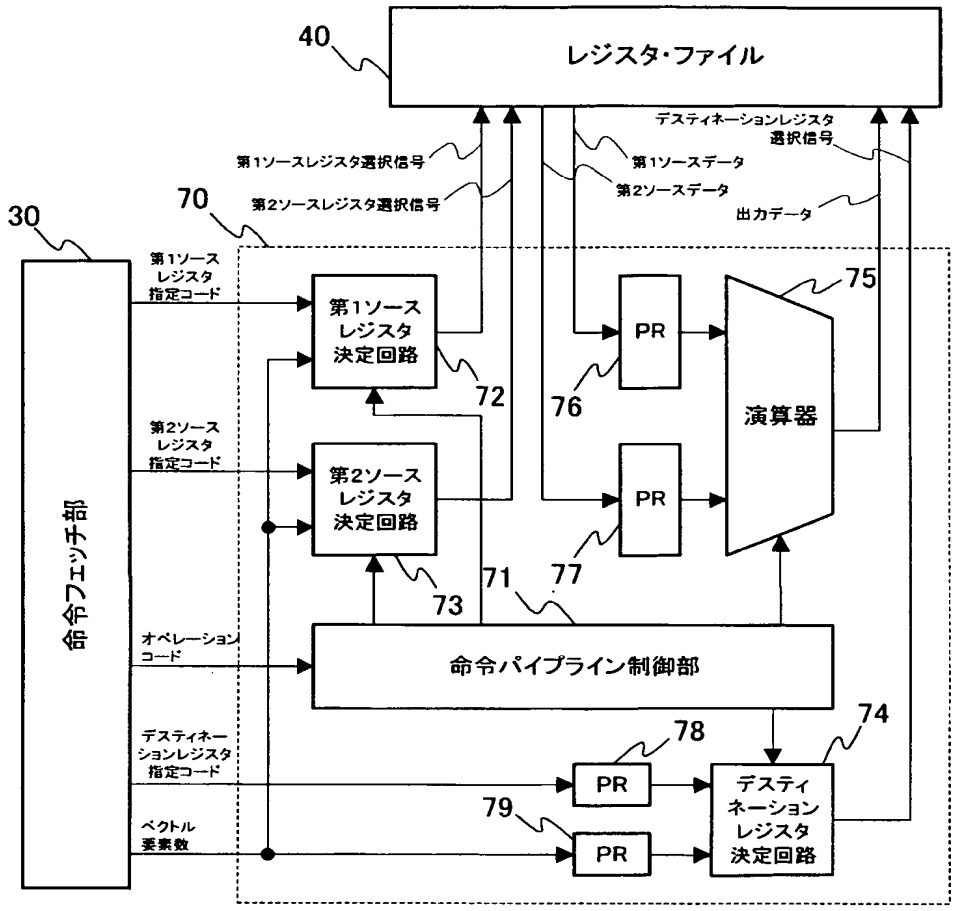
(c) 演算命令の形式

オペレーションコード	ベクトル要素数	デスティネーションレジスタ指定コード	第1ソースレジスタ指定コード	第2ソースレジスタ指定コード

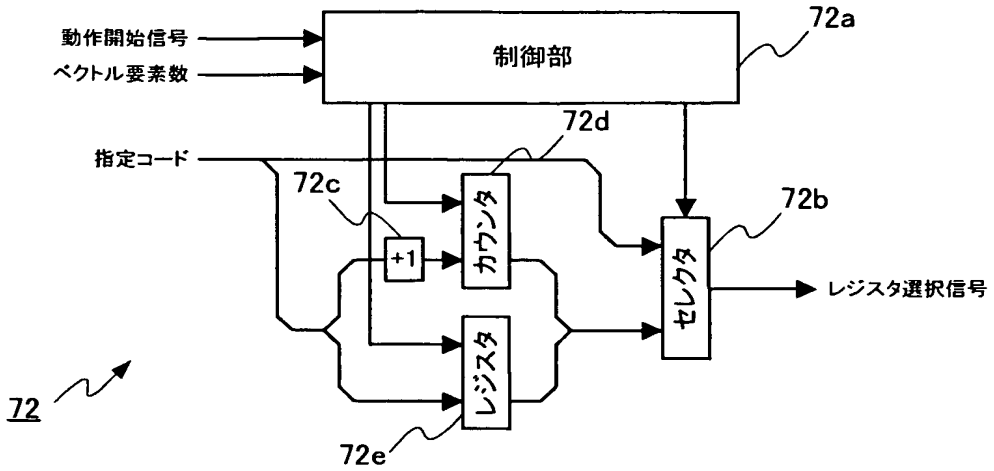
【図８】

レジスタ	コード	レジスタ	コード
R0	00000	R16	10000
R1	00001	R17	10001
R2	00010	R18	10010
R3	00011	R19	10011
R4	00100	R20	10100
R5	00101	R21	10101
R6	00110	R22	10110
R7	00111	R23	10111
R8	01000	R24	11000
R9	01001	R25	11001
R10	01010	R26	11010
R11	01011	R27	11011
R12	01100	R28	11100
R13	01101	R29	11101
R14	01110	R30	11110
R15	01111	R31	11111

【図９】


40
レジスタ・ファイル
デスティネーションレジスタ選択信号
第1ソースレジスタ選択信号
第1ソースデータ
第2ソースレジスタ選択信号
第2ソースデータ
出力データ
30
70
第1ソースレジスタ指定コード
第1ソースレジスタ決定回路
72
PR
76
75
演算器
第2ソースレジスタ指定コード
第2ソースレジスタ決定回路
PR
77
71
73
命令フェッチ部
オペレーションコード
命令パイプライン制御部
74
78
デスティネーションレジスタ指定コード
PR
デスティネーションレジスタ決定回路
79
ベクトル要素数
PR

【図１０】



【図１１】

ベクトル要素数	行われる演算
000	要素数１の演算（スカラ演算）
001	要素数２の演算
010	要素数３の演算
011	要素数４の演算
100	要素数５の演算
101	要素数６の演算
110	要素数７の演算
111	要素数８の演算

【書類名】　　要約書

【要約】

【課題】　ベクトルレジスタを用いたベクトル演算を効率的に行うこと。

【解決手段】　リングバッファをなすベクトルレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、１つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むことができる。したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【選択図】　　図５

特願２００３－０９２３７１

出　願　人　履　歴　情　報

識別番号　　[０００００２３６９]

１．変更年月日　　１９９０年　８月２０日
[変更理由]　　新規登録
住　所　　東京都新宿区西新宿２丁目４番１号
氏　名　　セイコーエプソン株式会社